LAPORAN PRAKTIKUM PEKAN 6

ALGORITMA DAN PEMROGRAMAN

PERULANGAN WHILE AND DO-WHILE

Disusun oleh:

Thaariq Salam

2511532022

Dosen Pengampu: Dr. Wahyudi S.T.M.T

Asisten Praktikum: Rahmad Dwirizki Olders

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga laporan praktikum mata kuliah Algoritma dan Pemrograman dengan topik "Perulangan *while* dan *do-while"* ini dapat disusun dan diselesaikan dengan baik. Laporan ini merupakan bagian dari pemenuhan tugas akademik pada Praktikum Pekan ke-6.

Topik Perulangan *while* dan *do-while* merupakan salah satu konsep dasar dalam pemrograman yang sangat penting untuk dipahami, karena memungkinkan eksekusi perintah secara berulang dengan kontrol yang efisien. Melalui praktikum ini, mahasiswa diharapkan mampu memahami struktur perulangan *while* dan *do-while*, serta mengimplementasikannya dalam berbagai skenario pemrograman yang membutuhkan iterasi.

Penyusun menyadari bahwa tersusunnya laporan ini tidak lepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, ucapan terima kasih kami sampaikan kepada:

1. Bapak Dr. Wahyudi, S.T., M.T., selaku Dosen Pengampu mata kuliah Algoritma dan Pemrograman.
2. Saudara Rahmad Dwirizki Olders, selaku Asisten Praktikum, atas bimbingan dan arahannya selama pelaksanaan praktikum.

Penyusun berharap laporan ini tidak hanya menjadi bentuk pertanggungjawaban akademik, tetapi juga dapat menjadi referensi yang bermanfaat bagi pembaca dalam memahami konsep perulangan *while* dan *do-while*. Kritik dan saran yang membangun sangat kami harapkan demi penyempurnaan laporan di masa mendatang.

Padang, 05 November 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam pemrograman, kemampuan untuk mengulang serangkaian instruksi merupakan aspek fundamental yang memastikan efisiensi dan fleksibilitas kode. Setelah menguasai struktur perulangan *for* yang ideal untuk jumlah iterasi yang telah diketahui, penting untuk mempelajari perulangan berbasis kondisi, yaitu *while loop* dan *do-while loop*. Struktur perulangan ini memungkinkan eksekusi blok kode secara berulang selama kondisi pengujian tertentu bernilai true.

Pemahaman terhadap *while* dan *do-while* menjadi sangat penting karena perulangan ini digunakan dalam skenario di mana jumlah pengulangan tidak diketahui sebelumnya atau bergantung pada input dinamis saat program berjalan. Contoh aplikasinya meliputi validasi input pengguna, pemrosesan data hingga batas tertentu (seperti *sentinel loop*). Dengan menguasai materi ini, diharapkan dapat membangun fondasi logika pemrograman yang kuat dan mampu merancang solusi yang cerdas dan adaptif terhadap variasi data input..

1.2 Tujuan

Praktikum Pekan 6 ini bertujuan untuk memperkuat pemahaman mahasiswa terhadap konsep perulangan berbasis kondisi. Adapun tujuan khusus dari praktikum ini adalah sebagai berikut:

1. Memahami prinsip dasar dan mekanisme pengujian kondisi pada struktur perulangan while (uji di awal) dan do-while (uji di akhir).

2. Mampu membedakan secara fundamental antara while dan do-while serta mengaplikasikannya secara tepat berdasarkan kebutuhan, khususnya untuk menjamin eksekusi minimum satu kali pada do-while.
3. Mampu mengimplementasikan while *loop* sebagai pengontrol sesi permainan dan sentinel *loop* yang kondisinya dipicu oleh input khusus.
4. Meningkatkan kemampuan logika dan analisis dalam merancang solusi algoritma yang efisien dan adaptif terhadap variasi input data melalui penggunaan perulangan berbasis kondisi.

1.3 Manfaat

Praktikum Pekan 6 mengenai while *loop* dan do-while *loop* memberikan kontribusi signifikan dalam pengembangan kemampuan teknis dan logis mahasiswa. Manfaat utamanya adalah:

1. Peningkatan kemampuan berpikir terstruktur:

   Mahasiswa dilatih untuk merancang solusi pemrograman yang efisien dalam mengolah data yang membutuhkan perulangan tak terbatas atau berulang hingga kondisi spesifik terpenuhi.
2. Penguasaan kontrol alur fleksibel:

   Penguasaan atas struktur while dan do-while merupakan fondasi penting dalam membangun program yang mampu menangani proses interaktif dan validasi input secara sistematis dan optimal.
3. Kesiapan aplikasi nyata:

   Mahasiswa tidak hanya belajar mengeksekusi perintah, tetapi juga memahami bagaimana mengelola alur logika permainan, *loop* validasi, dan *loop* yang dikontrol bendera, yang merupakan keterampilan kunci dalam aplikasi berbasis interaksi pengguna.

   .

BAB II

PEMBAHASAN

2.1 Program Do-While

2.1.1 Kode program

Kode program dan output hasilnya:

```java
package pekan6_2511532022;

import java.util.Scanner;

public class doWhile_2511532022 {
    public static void main (String[]args) {
        Scanner console =new Scanner (System.in);
        String phrase;
        do {
            System.out.print("Input Password: ");
            phrase = console.next();
        } while (!phrase.equals("abcd"));
    }
}
```

```
Problems  Javadoc  Declaration  Console  Git Sta
<terminated> doWhile_2511532022 [Java Application] C:\Users\user\.p2\p
Input Password: zbcd
Input Password: yyyy
Input Password: abcd
```

Gambar 2.1

2.1.2 Langkah kerja

Secara umum, program ini dirancang untuk meminta password sampai benar. Karena ini adalah *do-while loop*, cara kerjanya cukup santai:

1. Program akan langsung menjalankan instruksi di dalam blok *do*, yaitu menampilkan "Input Password:" dan menunggu kita mengetik, tanpa perlu mengecek apapun di awal.
2. Setelah kita memasukkan input, barulah program mengecek kondisi *while* (*!phrase.equals*("abcd")).
3. Kalau *password*-nya salah, dia akan mengulang lagi;

4. Kalau benar, *loop* langsung berhenti.

2.1.3 Analisis hasil

Program ini merupakan implementasi yang sangat efektif dari *do-while loop,* yang secara struktural dikenal sebagai *post-test loop*. Keunggulan utama dari *do-while* adalah kemampuannya memberikan jaminan eksekusi minimal satu kali pada blok kode sebelum kondisinya diuji [1]. Hal ini sangat ideal untuk kasus validasi input dan otentikasi password, karena program harus meminta input setidaknya satu kali agar ada sesuatu yang bisa divalidasi. Perulangan akan terus berlanjut (kondisi *true*) selama password yang dimasukkan masih belum sama dengan "abcd".

2.2 Program Game Penjumlahan

2.2.1 Kode program

Praktikum ini memperkenalkan konsep perulangan bersarang (*Nested Loops*), mengatasi keterbatasan perulangan tunggal dengan memungkinkan iterasi di dalam iterasi. Struktur kontrol ini esensial untuk tugas-tugas yang memerlukan pemrosesan data dua dimensi atau pembuatan pola terstruktur, menjamin setiap elemen dalam suatu dimensi (kolom) diproses secara penuh untuk setiap elemen pada dimensi lainnya (baris)[1].

```
1  package pekan6_2511532022;
2  import java.util.Scanner;
3  import java.util.Random;
4
5  public class GamePenjumlahan_2511532022 {
6      public static void main (String[]args) {
7          Scanner console = new Scanner(System.in);
8          Random rand= new Random ();
9          // play until user gets 3 wrong
10         int points = 0;
11         int wrong = 0;
12         while (wrong < 3) {
13             int result = play(console, rand);
14             if (result > 0) {
15                 points++;
16             } else {
17                 wrong++;
18             }
19         }
20         System.out.println("You earned " + points + " total points.");
21     }
```

```
22    public static int play(Scanner console , Random rand) {
23        int operands = rand.nextInt(4) + 2;
24        int sum = rand.nextInt(10) + 1;
25        System.out.print(sum);
26        for (int i = 2; i<= operands; i++) {
27            int n = rand.nextInt(10)+ 1;
28            sum += n;
29            System.out.print(" + " + n);
30        }
31        System.out.print(" = ");
32
33        //read user's guess and report whether it was correct
34        int guess = console.nextInt();
35        if (guess == sum) {
36            return 1;
37        } else {
38            System.out.println("Wrong! The answer was " + sum);
39            return 0;
40        }
41    }
42 }
43
```

Gambar 2.2

2.2.2 Langkah kerja

1. Program dimulai dengan menyetel tries (percobaan) dan sum (total lemparan) ke nol.
2. *Loop* utama akan terus berjalan selama total lemparan dadu belum mencapai angka 7.
3. Di setiap putaran, program akan "melempar" dua dadu secara acak (menghasilkan angka 1 sampai 6), menjumlahkannya, dan menampilkan hasilnya ke layar, sambil menaikkan hitungan tries.
4. Begitu totalnya kebetulan 7, perulangan berhenti dan program akan memberitahu kita berapa total percobaan yang dibutuhkan.

2.2.3 Analisis hasil

```
9 + 9 = 18
4 + 7 = 11
1 + 10 + 10 + 4 + 4 = 29
3 + 7 + 6 + 9 = 20
Wrong! The answer was 25
1 + 5 + 9 = 8
Wrong! The answer was 15
1 + 8 + 7 + 9 + 7 = 7
Wrong! The answer was 32
You earned 3 total points.
```

Gambar 2.3

Program ini adalah contoh penggunaan klasik dari *while loop* untuk memecahkan masalah dengan jumlah iterasi yang tidak pasti (indefinite *loop*s). Sebagai *pre-test loop*, perulangan *while* terus menguji kondisi *sum != 7* di awal setiap putaran.

Hal ini menunjukkan fleksibilitas *while* karena durasi perulangan tidak didasarkan pada perhitungan yang pasti (seperti *for loop*), melainkan sepenuhnya bergantung pada hasil acak yang dihasilkan di dalam *loop*. Ini merupakan skenario umum dalam simulasi probabilitas, di mana mekanisme pengubah kondisi (*sum* yang dihasilkan secara acak) harus ada agar *loop* dapat berhenti.

2.3 Program Lempar Dadu

2.3.1 Kode program

```
package pekan6_2511532022;
import java.util.Random;

public class Lempardadu_2511532022 {
    public static void main (String[]args) {
        Random rand= new Random ();
        int tries = 0;
        int sum = 0;
        while (sum != 7) {
            //roll the dice once
            int dadu1 = rand.nextInt(6)+ 1;
            int dadu2 = rand.nextInt(6)+ 1;
            sum = dadu1 + dadu2;
            System.out.println(dadu1 + " + " + dadu2 + " = " + sum);
            tries++;
        }
        System.out.println("You won after " + tries + " tries!");
    }
}
```

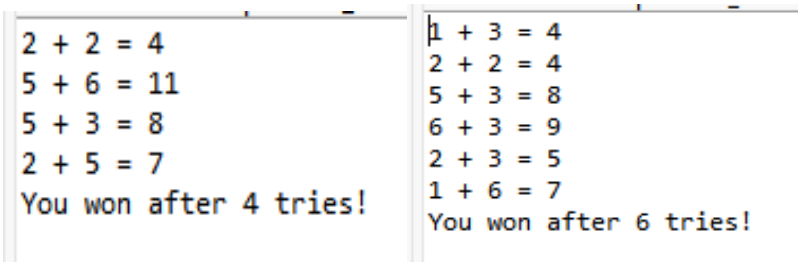
Gambar 2.4

2.3.2 Langkah kerja

1. Di program ini, kita menggunakan variabel running yang disetel true sebagai "tombol on" perulangan.
2. Selama running masih true, loop akan berjalan terus.
3. Di setiap putaran, program menghitung counter, lalu bertanya kepada pengguna, "Apakah lanjut (ya / tidak?)".
4. Kalau kita menjawab "tidak" (tidak peduli huruf besar atau kecil), maka variabel running diubah menjadi false oleh pernyataan if, dan perulangan pun berhenti.
5. Setelah loop selesai, program menampilkan total perulangan yang sudah dilakukan nilai 5.

2.3.3 Analisis hasil

Program ini berhasil mengaplikasikan teknik perulangan berbasis bendera (*Flag Controlled Loop).* Berbeda dengan perulangan berbasis hitungan, *while loop* di sini dikontrol oleh status variabel *boolean* (*running*) yang dapat dimanipulasi oleh logika internal, yaitu interaksi dari pengguna. Ini adalah metode yang sangat efisien dan interaktif untuk membuat program yang berjalan berulang kali, di mana pengguna secara eksplisit memutuskan kapan perulangan harus diakhiri dengan mengubah nilai *flag* dari *true* menjadi *false*. Berikut 2 sampel output program:



Gambar 2.5 dan 2.6

2.4 Program Sentinel Loop

2.4.1 Kode program

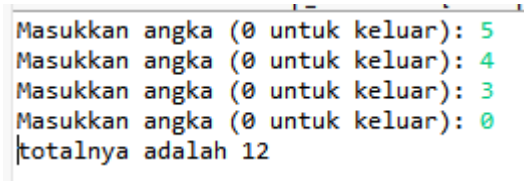
```java
package pekan6_2511532022;
import java.util.Scanner;
public class SentinelLoop_2511532022 {

    public static void main (String[]args) {
        Scanner console =new Scanner (System.in);
        int sum=0;
        int number=12; // "dummy value", anything but 0

        while (number != 0) {
            System.out.print("Masukkan angka (0 untuk keluar): ");
            number = console.nextInt();
            sum = sum + number;
        }
        System.out.println("totalnya adalah " + sum);
    }
}
```

Gambar 2.7

2.4.2 Langkah kerja

1. Persiapkan *sum* dengan nilai 0 dan *number* dengan nilai *dummy* (misalnya 12), agar *loop* bisa langsung jalan.
2. Perulangan utama akan terus berjalan selama angka yang dimasukkan pengguna bukan nol.
3. Di setiap putaran, kita diminta memasukkan angka, dan angka tersebut langsung ditambahkan ke sum.
4. Begitu kita memasukkan angka nol, program menganggap itu adalah sinyal berhenti (*sentinel*), dan *loop* langsung berakhir, lalu total penjumlahannya pun ditampilkan.

2.4.3 Analisis hasil



Gambar 2.8

Program ini menunjukkan konsep *Sentinel Loop*, yaitu perulangan yang dipicu untuk berhenti oleh *input* yang memiliki nilai khusus (*sentinel value*) yang tidak mungkin menjadi bagian dari data yang valid, dalam hal ini adalah 0 [1]. *Sentinel Loop* ini sangat berguna untuk mengakumulasi atau memproses serangkaian data yang jumlahnya tidak diketahui sebelumnya. Variabel kontrol (*number*) harus diberi nilai *dummy* (seperti 12) di awal untuk memastikan kondisi *while* (*number != 0*) bernilai *true* pada pengujian pertama, sehingga *loop* dapat dimulai dan pengguna dapat memasukkan data pertamanya.

2.5 Program Game Penjumlahan

2.5.1 Kode program

```java
package pekan6_2511532022;
import java.util.Scanner;
import java.util.Random;

public class GamePenjumlahan_2511532022 {
    public static void main (String[]args) {
        Scanner console = new Scanner(System.in);
        Random rand= new Random ();
        // play until user gets 3 wrong
        int points = 0;
        int wrong = 0;
        while (wrong < 3) {
            int result = play(console, rand);
            if (result > 0) {
                points++;
            } else {
                wrong++;
            }
        }
        System.out.println("You earned " + points + " total points.");
    }
    public static int play(Scanner console , Random rand) {
        int operands = rand.nextInt(4) + 2;
        int sum = rand.nextInt(10) + 1;
        System.out.print(sum);
        for (int i = 2; i<= operands; i++) {
            int n = rand.nextInt(10)+ 1;
            sum += n;
            System.out.print(" + " + n);
        }
        System.out.print(" = ");

         //read user's guess and report whether it was correct
         int guess = console.nextInt();
         if (guess == sum) {
             return 1;
         } else {
             System.out.println("Wrong! The answer was " + sum);
             return 0;
         }
     }
}
```

Gambar 2.9

2.5.2 Langkah kerja

5. Program dimulai dengan menyetel tries (percobaan) dan sum (total lemparan) ke nol.
6. *Loop* utama akan terus berjalan selama total lemparan dadu belum mencapai angka 7.
7. Di setiap putaran, program akan "melempar" dua dadu secara acak (menghasilkan angka 1 sampai 6), menjumlahkannya, dan menampilkan hasilnya ke layar, sambil menaikkan hitungan tries.
8. Begitu totalnya kebetulan 7, perulangan berhenti dan program akan memberitahu kita berapa total percobaan yang dibutuhkan.

2.5.3 Analisis hasil

```
9 + 9 = 18
4 + 7 = 11
1 + 10 + 10 + 4 + 4 = 29
3 + 7 + 6 + 9 = 20
Wrong! The answer was 25
1 + 5 + 9 = 8
Wrong! The answer was 15
1 + 8 + 7 + 9 + 7 = 7
Wrong! The answer was 32
You earned 3 total points.
```

Gambar 2.10

Program ini adalah contoh penggunaan klasik dari *while loop* untuk memecahkan masalah dengan jumlah iterasi yang tidak pasti (indefinite *loop*s). Sebagai *pre-test loop*, perulangan *while* terus menguji kondisi *sum != 7* di awal setiap putaran. Hal ini menunjukkan fleksibilitas while karena durasi perulangan tidak didasarkan pada perhitungan yang pasti (seperti *for loop*), melainkan sepenuhnya bergantung pada hasil acak yang dihasilkan di dalam *loop*. Ini merupakan skenario umum dalam simulasi probabilitas, di mana mekanisme pengubah kondisi (*sum* yang dihasilkan secara acak) harus ada agar *loop* dapat berhenti.

BAB III

PENUTUP

3.1 Kesimpulan

Praktikum Pekan 6 telah berhasil memberikan pemahaman yang mendalam tentang struktur perulangan berbasis kondisi, yaitu while dan do-while, serta berbagai aplikasinya.

1. Sifat Dasar dan Perbedaan Krusial: Kedua perulangan ini dikenal sebagai condition-controlled loops dan sangat cocok untuk skenario dengan jumlah iterasi yang tidak pasti. Perbedaan utamanya terletak pada lokasi pengujian kondisi: while adalah pre-test loop (uji di awal), yang berarti dapat dieksekusi nol kali, ideal untuk Sentinel Loop atau simulasi acak. Sementara itu, do-while adalah post-test loop (uji di akhir), yang menjamin blok kode dieksekusi minimal satu kali, menjadikannya pilihan utama untuk validasi input dan otentikasi password [1].
2. Penerapan Lanjutan: Melalui implementasi Flag Controlled Loop dan Sentinel Loop, dapat disimpulkan bahwa while loop sangat fleksibel. Kondisi berhentinya dapat dikontrol oleh variabel boolean (running) yang diubah oleh pengguna, atau oleh nilai penanda (0), membuktikan kemampuannya untuk mengelola alur program berdasarkan interaksi dinamis.
3. Modularitas: Perulangan while juga dapat berfungsi sebagai kontrol sesi utama (Game Loop), di mana logikanya bekerja secara modular dengan memanggil fungsi lain (seperti play()) yang mengintegrasikan for loop di dalamnya, menghasilkan program yang terstruktur dan mudah dikelola.

DAFTAR PUSTAKA

[1] P. Deitel dan H. Deitel, Java How to Program, Early Objects, 11th ed. Boston, MA: Pearson, 2018